

Programación 1



Tema 2

Tipos de Datos Simples



Departamento de Ciencia de la
Computación e Inteligencia Artificial

Objetivos / Competencias

1. Comprender el uso de datos en un programa
2. Conocer los tipos de datos simples de un lenguaje de programación
3. Aprender a manejar, leer e imprimir tipos de datos simples en lenguaje C (C++)

Índice

1. Tipos de datos en un programa



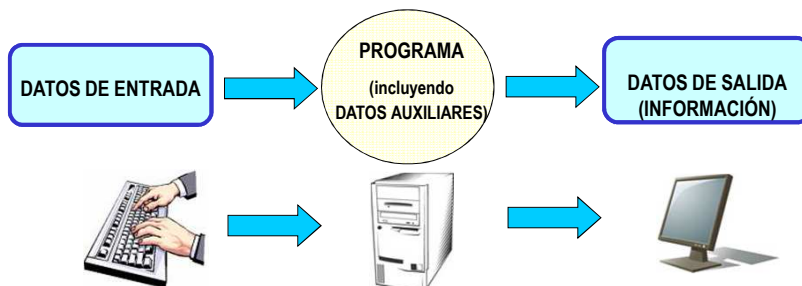
2. Datos variables y constantes
3. Manejar variables y constantes en un programa
4. Sentencia de asignación
5. Expresiones aritméticas y lógicas
6. Sentencias de entrada y salida de datos
7. Estructura general de un programa
8. Fuentes de información

Datos en un Programa

Dato = hecho o valor a partir del cual se puede inferir una conclusión (información)

Datos en un programa = datos con los que opera una computadora

- ◆ Los **datos de entrada** constituyen un punto de partida para obtener conocimiento (**datos de salida**)
- ◆ El programa también puede necesitar **datos auxiliares** (internos) para obtener el resultado



Ejemplo de datos en un programa

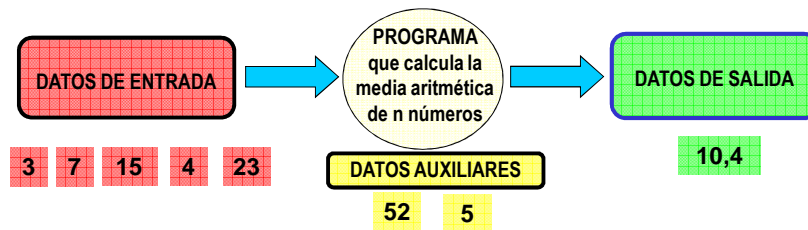
Programa : calcula la media aritmética de n números

Datos de Entrada : n números cualesquiera

Datos de Salida : la media aritmética de los n números

Datos Auxiliares :

- la suma de los números
- la cantidad de números



Programación 1. Dto. CCIA. Curso 2013-14

P-5

Tipos de datos en un programa

Tipo de dato = Valores + Operaciones

- Conjunto de **valores** que puede tomar un dato en el programa
 - Si se le intenta dar un valor fuera del conjunto entonces puede producirse un error
- Conjunto de **operaciones** que se pueden definir sobre los datos

Programación 1. Dto. CCIA. Curso 2013-14

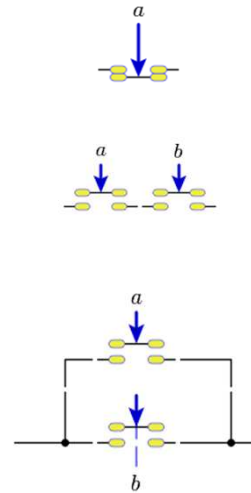
P-6

Ejemplo de tipo de dato

Tipo de Dato **Booleano**

- Valores = { **true**, **false** }
- Operaciones = { **and**, **or**, **not** }

a	b	not a	a and b	a or b
false	false	true	false	false
true	false	false	false	true
false	true		false	true
true	true		true	true



Programación 1. Dto. CCIA. Curso 2013-14

P-7

Tipo de dato simple

- Son tipos elementales que no se derivan de otros tipos
- Cada valor concreto del tipo de dato simple viene especificado por un **literal**
 - Por ejemplo, los literales enteros pueden expresarse:
 - En decimal (base 10) : 255
 - En octal (base 8) : 0377 ($3 \cdot 8^2 + 7 \cdot 8^1 + 7 = 255$)
 - En hexadecimal (base 16): 0xff ($15 \cdot 16^1 + 15 = 255$)

Programación 1. Dto. CCIA. Curso 2013-14

P-8

Tipos de datos simples predefinidos en C

Tipo	Significado	Bytes
carácter		
<code>char</code>	carácter	1
<code>unsigned char</code>	carácter sin signo	1
numérico		
entero		
<code>int</code>	entero	2-4
<code>short</code>	entero corto	2
<code>long</code>	entero largo	4
<code>long long</code>	entero largo	8
<code>unsigned</code>	entero sin signo	2-4
<code>unsigned short</code>	entero corto sin signo	2
<code>unsigned long</code>	entero largo sin signo	4
<code>unsigned long long</code>	entero largo sin signo	8
real		
<code>float</code>	coma flotante (número real)	4
<code>double</code>	coma flotante de doble precisión	8
<code>long double</code>	coma flotante de doble precisión extendida	16
booleano		
<code>bool</code>	Booleano	1



El tipo **bool** no existe en el estándar ANSI C y es emulado con el tipo **int** (cero es valor false, y distinto de cero es valor true)

Nosotros utilizaremos el tipo **bool** de C++ y del estándar C99

Programación 1. Dto. CCIA. Curso 2013-14

P-9

Valores de tipos de datos en C

Tipo	Bytes	Valores	precisión
carácter	1	Alfabéticos: 'a', 'b', ... 'z' 'A', 'B', ... 'Z' Dígitos: '0', '1', '2', ... '9' Especiales: '+', '-', '/', '=', '(', ...	
short	2	-32.767..32.767	
int	4	-2.147.483.647..2.147.483.647	
float	4	Aprox. 10^{-38} .. 10^{38}	7 dígitos
double	8	Aprox. 10^{-308} .. 10^{308}	15 dígitos
bool	1	true, false	

Programación 1. Dto. CCIA. Curso 2013-14

P-10

Tipos de datos enumerados


Generalmente los lenguajes de programación tienen tipos de datos predefinidos y además posibilitan al usuario definir sus propios tipos de datos

En el lenguaje C

- El usuario puede definir tipos de datos enumerados compuestos por un conjunto de identificadores que representan un valor entero
- No hay formato de impresión para estos tipos. El primer elemento tiene asociado el valor 0, el segundo el valor 1 y así sucesivamente

```
enum T_DiaSemana {lunes, martes, miercoles, jueves, viernes, sabado, domingo};  
enum T_Color_Primary {rojo, verde, azul};
```

Índice

1. Tipos de datos en un programa
2. **Datos variables y constantes** 
3. Manejar variables y constantes en un programa
4. Sentencia de asignación
5. Expresiones aritméticas y lógicas
6. Sentencias de entrada y salida de datos
7. Estructura general de un programa
8. Fuentes de información

Variables y Constantes en un programa

Características comunes:

- ◆ Permiten representar datos en un programa
- ◆ Constituyen un espacio de memoria reservado para almacenar un valor de un tipo de dato
- ◆ Se identifican con un nombre

Se diferencian en ...

- ◆ El valor de una variable puede cambiar a lo largo del programa
- ◆ El valor de una constante nunca cambia en el programa

Programación 1. Dto. CCIA. Curso 2013-14

P-13

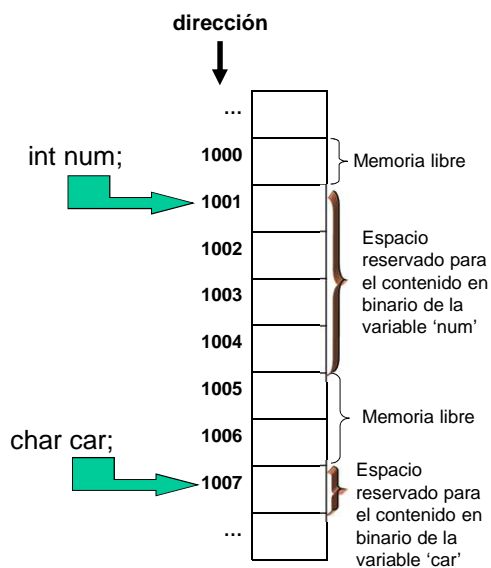
Representación en memoria de las variables

◆ La memoria consiste en una lista de posiciones numeradas (bytes)

◆ Una variable representa una porción de memoria compuesta por un número consecutivo de bytes

◆ Una variable en memoria viene determinada por:

- La **dirección** en la memoria que proporciona la ubicación del **primer byte** dedicado a esa variable
- El **tipo** determina **cuántos bytes** de memoria requiere la variable



Programación 1. Dto. CCIA. Curso 2013-14

P-14

Ejemplo de variables y constantes

Un club de fútbol X propietario de un estadio Y, necesita calcular la recaudación de cada uno los partidos disputados en su estadio, teniendo en cuenta que pone a la venta **tres tipos de entrada** dependiendo del lugar en la grada del asiento seleccionado por el aficionado: **entrada de fondo** (gradas de detrás de las porterías), **entrada general** (gradas laterales sin techo) y **entrada preferente** (gradas laterales con techo). Durante toda la temporada, el precio de una entrada de fondo es la mitad que el de una entrada general y el precio de una entrada preferente es el doble de una entrada general. En cada partido, el club de fútbol fija un precio para la entrada general y además establece unos **descuentos** para toda la temporada para los **niños** (el 80%) y para los **pensionistas** (el 50%).

Para calcular la recaudación de cada partido de la temporada ...

- Para almacenar el precio de la entrada general ... Variable real
- Para almacenar el descuento para los niños ... Constante
- Para almacenar el número de entradas preferentes vendidas ... Variable entera
- ¿y para almacenar el tipo de entrada vendida?...
- ¿y para almacenar si a un aficionado se le aplica descuento o no?...
- ¿y para almacenar el precio de una entrada preferente?...

Programación 1. Dto. CCIA. Curso 2013-14

P-15

Identificadores de variables y constantes

Notaciones muy extendidas en la mayoría de programadores:

1. Las **variables en minúsculas** y las **constantes en mayúsculas**
2. Con identificadores compuestos por varias palabras:
 - Todo en minúsculas, separando las palabras con el carácter subrayado
`nombre_alumno`
 - Todo en mayúsculas, separando las palabras con el carácter subrayado
`NOMBRE_ALUMNO`
 - Todo en minúsculas excepto las iniciales de cada palabra
`NombreAlumno`
 - Utilización de abreviaturas con la misma longitud
`nom_alu`



Es muy importante no cambiar arbitrariamente de **notación** y seguir sólo una de ellas para mantener una coherencia en nuestros programas y facilitar la legibilidad y la comprensión de los mismos

Programación 1. Dto. CCIA. Curso 2013-14

P-16

Identificadores en un programa

Un **identificador** es un nombre que utiliza el programador para referenciar los datos y otros elementos del programa

Reglas generales de construcción de identificadores:

1. Debe resultar significativo
2. No puede coincidir con palabras reservadas propias del lenguaje de programación
3. La longitud no debe ser excesivamente larga
4. Deben comenzar por un carácter alfabético o el símbolo de subrayado y pueden contener caracteres alfabéticos, dígitos y el símbolo de subrayado
5. No se acentúan
6. Según el lenguaje de programación podrá ser utilizado indistintamente o no, en mayúsculas o en minúsculas



Los lenguajes C y C++ son **sensibles** a mayúsculas y minúsculas

Programación 1. Dto. CCIA. Curso 2013-14

P-17

Ejemplos de identificadores



Identificadores **correctos**:

- distancia
- distancia_euclidea
- _fecha
- fechaNacimiento
- NUMERO_PI
- numero1
- numero_2



Identificadores **incorrectos**:

- distancia-euclidea
- 3libros
- Numero\$1
- Mas_preguntas?
- número



Los siguientes **identificadores** son distintos en lenguaje C:

Color_coche color_coche COLOR_COCHE color_Coche Color_Coche

Programación 1. Dto. CCIA. Curso 2013-14

P-18

Índice

1. Tipos de datos en un programa
2. Datos variables y constantes
3. **Manejar variables y constantes en un programa**
4. Sentencia de asignación
5. Expresiones aritméticas y lógicas
6. Sentencias de entrada y salida de datos
7. Estructura general de un programa
8. Fuentes de información



¿Cómo se manejan las variables y constantes?

Paso 1: hay que declararlas

- El programador debe darles un nombre y determinar su tipo de dato para que el compilador reserve el espacio de memoria necesario para almacenar un valor de dicho tipo de dato

Paso 2: hay que inicializarlas

- El programador debe asignar un primer valor antes de que sea utilizada

Paso 3: hay que utilizarlas

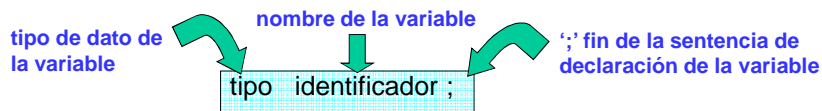
- El programador las debe utilizar en los lugares del programa (sentencias) que le sean permitidos según las reglas sintácticas que establece el lenguaje de programación empleado

Paso 4: hay que destruirlas

- El compilador libera el espacio de memoria previamente reservado
- Normalmente no es labor del programador realizar este paso, pero debe tener en cuenta cuándo se produce para no utilizar variables y constantes una vez destruidas

Declaración de variables en lenguaje C

Hay que asociar un tipo de dato a la variable para que en ésta se pueda almacenar cualquier valor de ese tipo de dato



```
char letra_dni; // variable para almacenar la letra del dni de cualquier persona
int paginas; // variable para almacenar el n° de páginas de cualquier libro
float sueldo; // variable para almacenar el sueldo de cualquier persona
bool aprobado; // variable para almacenar si un alumno ha aprobado o no una asignatura
```

Programación 1. Dto. CCIA. Curso 2013-14

P-21

Declaración e inicialización de constantes en C / C++

En lenguaje C

Indica que se va a declarar un valor constante

nombre de la constante

valor de la constante

`#define identificador valor`

```
#define HORAS_DIA 24 // constante que almacena el n° de horas de un día
```

En lenguaje C++

palabra reservada de C++

tipo de dato de la constante

nombre de la constante

'=' operador de asignación

',' fin de sentencia

`const tipo identificador = valor ;`

```
const int HORAS_DIA = 24; // constante que almacena el n° de horas de un día
```

Programación 1. Dto. CCIA. Curso 2013-14

P-22

Inicialización de variables en lenguaje C

Mediante la **sentencia de asignación** :



En lenguaje C

```
letra_dni = 'A'; // almacena el carácter 'A' en la variable letra_dni
paginas = 365; // almacena el número 365 en la variable paginas
sueldo = 1000.20; // almacena el número real 1000.20 en la variable sueldo
aprobado = true; // almacena el valor true en la variable aprobado
```



En C y C++ se permite **inicializar** variables en su declaración, por ejemplo: `int paginas = 365;`

Programación 1. Dto. CCIA. Curso 2013-14

P-23

Utilización de variables y constantes en lenguaje C

Una **variable** se utiliza ...

- en la parte izquierda de una sentencia de asignación
- en una expresión aritmética o lógica
- en sentencias de entrada y salida de datos

Una **constante** se utiliza ...

- en una expresión aritmética o lógica
- en sentencias de salida de datos




¿Por qué no una constante en la parte izquierda de una asignación?

¿Por qué no una constante en sentencias de entrada de datos?

Programación 1. Dto. CCIA. Curso 2013-14

P-24

Índice

1. Tipos de datos en un programa
2. Datos variables y constantes
3. Manejar variables y constantes en un programa
4. **Sentencia de asignación** 
5. Expresiones aritméticas y lógicas
6. Sentencias de entrada y salida de datos
7. Estructura general de un programa
8. Fuentes de información

Sintaxis de la sentencia de asignación



En lenguaje C

```
paginas_libroA = 430; // almacena el número 430 en la variable paginas_libroA, declarada previamente de tipo int  
sueldo = 35616.44; // almacena el número real 35616.44 en la variable sueldo, declarada previamente de tipo float
```

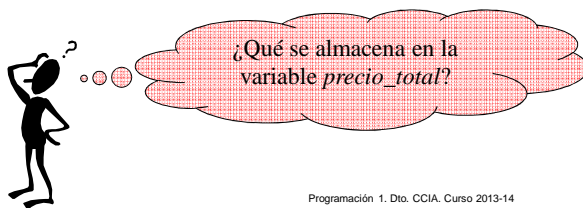
¿Cómo funciona la sentencia de asignación?

Paso 1: Se evalúa la parte derecha del operador de asignación

Paso 2: Se asigna el valor de la parte derecha a la variable de la parte izquierda del operador de asignación

En lenguaje C

```
// suponiendo que se han declarado previamente las variables relacionadas con precios de tipo float
precio_cocheA = 10500.00; // almacena el número 10500.00 en la variable precio_cocheA
precio_cocheB = 40200.00; // almacena el número 40200.00 en la variable precio_cocheB
precio_total = precio_cocheA + precio_cocheB;
```



Programación 1. Dto. CCIA. Curso 2013-14

P-27

Índice

1. Tipos de datos en un programa
2. Datos variables y constantes
3. Manejar variables y constantes en un programa
4. Sentencia de asignación
- 5. Expresiones aritméticas y lógicas**
6. Sentencias de entrada y salida de datos
7. Estructura general de un programa
8. Fuentes de información



Programación 1. Dto. CCIA. Curso 2013-14

P-28

Expresiones aritméticas y lógicas

Una **expresión** en un programa es una combinación de variables, constantes, operadores, paréntesis e identificadores de funciones, de cuya evaluación se obtiene un valor.

- Las expresiones se pueden escribir en cualquier lugar del programa en donde pueda utilizarse el valor que devuelven

Una **expresión aritmética** ...

- se construye con operadores aritméticos
- devuelve un valor numérico

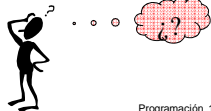
```
(x_rad * 360) / (2 * PI)
```

calcula los grados correspondientes al valor en radianes almacenado en la variable `x_rad`, utilizando la constante `PI`

Una **expresión lógica** ...

- se construye con operadores relacionales y lógicos
- pueden aparecer operadores aritméticos
- devuelve un valor booleano

```
(año modulo 4 == 0) AND (NOT (año modulo 100 == 0) OR (año modulo 400 == 0) )
```



Programación 1. Dto. CCIA. Curso 2013-14

P-29

Operadores en lenguaje C

operadores aritméticos	significado	tipos de operandos	tipo de resultado
+ - * /	suma, resta, multiplicación, división	numéricos enteros o reales	numérico entero o real
%	resto de división	enteros	entero
operadores relacionales			
< > <= >=	menor que, mayor que, menor o igual que, mayor o igual que	tipos simples	booleano
== !=	igual que, distinto de	tipos simples	booleano
operadores lógicos			
&&	AND lógico	booleano	booleano
	OR lógico	booleano	booleano
!	NOT lógico	booleano	booleano



Ten clara la diferencia entre el **operador de asignación** '=' y el operador relacional de **igualdad** '=='. Es habitual utilizar erróneamente el operador '=' en lugar de '==', lo que provoca errores difíciles de detectar

Con el **operador de división** '/', cuando los operandos son de tipo numérico entero, el resultado es la parte entera del cociente. Para obtener un resultado con decimales, alguno de los operandos debe ser de tipo numérico real.

Programación 1. Dto. CCIA. Curso 2013-14

P-30

Precedencia y Asociatividad de operadores

La **precedencia** o **prioridad** de un operador indica el orden en que se ejecutan las operaciones en una expresión que contiene distintos operadores

La **asociatividad** de un operador indica el orden en que se ejecutan las operaciones en una expresión que contiene operadores con la misma prioridad

Orden de precedencia de los operadores en lenguaje C

operadores	significado	asociatividad
- !	signo negativo de un número, NOT lógico	de derecha a izquierda
* / %	multiplicación, división, resto	de izquierda a derecha
+ -	suma, resta	de izquierda a derecha
< > <= >=	de relación	de izquierda a derecha
== !=	de igualdad	de izquierda a derecha
&&	AND lógico	de izquierda a derecha
	OR lógico	de izquierda a derecha



Es recomendable el uso de **paréntesis**:

- cuando tengamos alguna duda del orden de evaluación
- para hacer más legible la misma
- para modificar el orden de evaluación

Programación 1. Dto. CCIA. Curso 2013-14

P-31

Índice

1. Tipos de datos en un programa
2. Datos variables y constantes
3. Manejar variables y constantes en un programa
4. Sentencia de asignación
5. Expresiones aritméticas y lógicas
- 6. Sentencias de entrada y salida de datos**
7. Estructura general de un programa
8. Fuentes de información



Programación 1. Dto. CCIA. Curso 2013-14

P-32

Sentencias de Entrada y Salida de datos

- Las variables también pueden utilizarse en sentencias de entrada
- Las variables, constantes y en general las expresiones también pueden utilizarse en sentencias de salida
- Las **sentencias de entrada** permiten almacenar en variables datos que el usuario introduce por teclado
- Las **sentencias de salida** permiten visualizar datos en la pantalla



La entrada y salida puede estar asociada a distintas **fuentes y dispositivos**, tales como ficheros, impresoras, pantallas táctiles, ratón, etc.

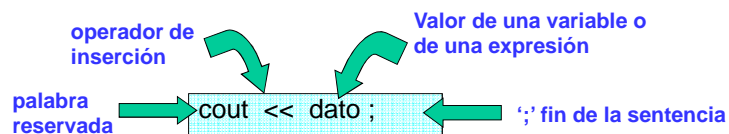
En esta asignatura, nosotros sólo usaremos en nuestros programas el **teclado y la pantalla** que suelen ser los dispositivos de entrada y salida por defecto.

Programación 1. Dto. CCIA. Curso 2013-14

P-33

Sentencia de salida en C++

- Permite escribir en pantalla cualquier combinación de valores de variables, constantes, expresiones y cadenas de texto



Ejemplos

```
cout << "El precio del ordenador portátil es de " << precio << " euros" << endl;  
cout << "el precio total es : " << (precio1 + precio2);  
cout << precio;  
cout << "esto es una cadena de texto sin salto a una nueva línea";  
cout << "esto es una cadena de texto con salto a una nueva línea\n";  
cout << endl;  
cout << "\n";
```



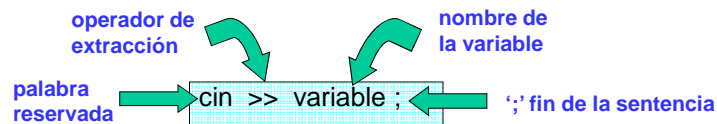
En lenguaje C para la salida de datos se utiliza la función de librería **printf()**, pero optamos por utilizar la sentencia **cout** de C++ porque es más sencilla de usar

Programación 1. Dto. CCIA. Curso 2013-14

P-34

Sentencia de entrada en C++

- ◆ Permite almacenar en variables valores introducidos por teclado



Ejemplos

```
cout << "Introduce tu edad:";
cin >> edad; // edad será una variable declarada de tipo int
cout << "introduce las notas de los dos parciales de prácticas:";
cin >> nota1 >> nota2; // nota1 y nota2 serán variables declaradas de tipo float o double
cout << "¿Deseas introducir más datos? (S/N) : ";
cin >> respuesta; // respuesta será una variable declarada de tipo char
```



En lenguaje C para la entrada de datos se utiliza la función de librería `scanf()`, pero optamos por utilizar la sentencia `cin` de C++ porque es más sencilla de usar

Índice

1. Tipos de datos en un programa
2. Datos variables y constantes
3. Manejar variables y constantes en un programa
4. Sentencia de asignación
5. Expresiones aritméticas y lógicas
6. Sentencias de entrada y salida de datos
- 7. Estructura general de un programa**
8. Fuentes de información



¿Qué tipo de programas debo ser capaz de hacer?

#directivas del preprocesador

Declaración de constantes

main() {

Declaración de variables:

de tipos simples

Cuerpo principal (sentencias ejecutables)

sentencias de Entrada y Salida

sentencias de asignación

}

```
#include <iostream>
```

```
using namespace std;
```

```
const float PI = 3.1415926;
```

Declaración e
inicialización
de una
constante

main() {

```
float area, radio;
```

Declaración de
dos variables

```
cout << "Introduce el radio del círculo:";
```

```
cin >> radio;
```

```
area = PI * radio * radio;
```

```
cout << "El área del círculo es:" << area;
```

```
cout << endl;
```

}



Todas las sentencias en C y C++ terminan con un **punto y coma**

Programación 1. Dto. CCIA. Curso 2013-14

P-37

Índice

1. Tipos de datos en un programa
2. Datos variables y constantes
3. Manejar variables y constantes en un programa
4. Sentencia de asignación
5. Expresiones aritméticas y lógicas
6. Sentencias de entrada y salida de datos
7. Estructura general de un programa
8. Fuentes de información



Programación 1. Dto. CCIA. Curso 2013-14

P-38

Bibliografía Recomendada

Fundamentos de Programación
Jesús Carretero, Félix García, y otros
Thomson-Paraninfo 2007. ISBN: 978-84-9732-550-9

▣ Capítulo 2 (Apartados 2.4)

▣ Capítulo 4 (Apartados 4.1; 4.2; 4.3; 4.4; 4.10)

Problemas Resueltos de Programación en Lenguaje C
Félix García, Alejandro Calderón, y otros
Thomson (2002) ISBN: 84-9732-102-2

▣ Capítulo 2 (Apartados 2.1; 2.2; 2.3)

Resolución de Problemas con C++
Walter Savitch
Pearson Addison Wesley 2007. ISBN: 978-970-26-0806-6

▣ Capítulo 2